intake, fluid output and the like, which were written on different paper forms by a nurse and later keyed into this common database. Similarly, when patients undergo testing, the test results were manually keyed into the common database and/or written on forms stored in the patient's paper file.

On Page 2, amend the third paragraph as follows:

However, every business requiring significant amounts of data acquisition and retrieval in its day-to-day business encounter similar problems. First and foremost, a significant amount of the field operative's time is required in filling out the corresponding paperwork, in which the potential for user error exists. Also, in systems using paper forms, the information must be ultimately transferred to an electronic database, which provides a second opportunity for user error. Clearly, it would be advantageous to [eliminate] reduce the number of user entries, thereby reducing the likelihood of error.

On Page 4, amend the seventh paragraph as follows:

It is another object of the present invention to provide [an] a user interface which is easily operated by using a touch pad which presents a scroll bar, rolling keys and icons for data entry.

On Page 5, amend the first paragraph as follows:

It is another object of the present invention is to provide [an] a user interface which allows for the scanning of bar codes to identify particular customers or products.

On Page 6, amend the second paragraph as follows:

[Fig. 7 illustrates] Figs. 7(a) and 7(b) illustrate the processing sequence of the handheld interface while displaying a patient input screen;

On Page 7, at line 2, insert the following paragraph:

"Fig. 14 illustrates the processing QUEUE sequence to move completed messages to the processing QUEUE;"

On Page 8, amend the third paragraph as follows:

During operation, the user enters data at the handheld interface 8, the data is transmitted to the communications server 12 and stored internally within the database 16. Data may also be entered directly into the communications server 12. Similarly, the user may request data previously submitted, in which case the communications server 12 accesses the corresponding database 16, through the command server 14, and transmits the necessary desired information to the requesting handheld interface 8. Throughout operation, a backup copy may be maintained within the master server 2 for every remote database 16. Additionally, the user may request, via the handheld interface 8, data stored within a remote input unit 4 other than the [than] data in the databases directly connected to the receiving communications server 12. In such a circumstance, the corresponding communications server 12 would accept the request from the handheld interface 8, determine that the desired information is stored within a remote database and request such information through the communications bus 6 from the communications server 12 containing the corresponding database. Thus, the communications bus 6 also allows data to be transmitted between communications servers 12 and between handheld interfaces 8 (such as when one doctor is requesting information from another doctor).

On Page 11, amend the third and fourth paragraphs as follows:

The manner by which these objects and functions [is] are achieved will become clear in connection with the following explanation of each module of the present invention.

Handheld Interface

Fig. 2 generally illustrates a block diagram of a handheld interface 8 having a display screen 30 which may be back-illuminated and which is controlled by a CPU 32 to display desired information thereon. The display screen 30 also functions as a touch pad and is sensitive to user contact. When contacted, the display screen 30 outputs a signal to the CPU 32 identifying the exact location of the contact therewith. The handheld interface 8 further includes a memory module 34 which stores software to control processing of the CPU 32 and which temporarily stores data received from, and transmitted to, the corresponding communications server 12. The CPU 32 controls an IR interface 36 to control data transmissions to and from the communications server 12. A bar code reader 37 is included to allow the user to enter customer or product information from a bar code, such as customer/patient ID and the like.

On Page 13, please amend the second full paragraph as follows:

As illustrated in Fig. 3c, the vitals input screen 60 (also referred to as the data I/O screen) displays current patient information in the patient field 62, such as the patient's name, social security number, status, and the date upon which the vitals were last updated. The vitals input screen 60 illustrates the last current vitals (data sets) as shown in the systolic field 46, diastolic field 48, pulse field 50, temperature field 52, and respiratory field 54 (collectively referred to as data fields). The touch screen 30 allows the user to activate a desired vital sign field by selecting a corresponding icon 46, 48, 50, 52, and 54, respectively. Located immediately above each icon is the current value for the corresponding vitals field. This current value is also displayed within three rolling keys along side of the icon (the rolling keys are designated by the reference numeral 56). The patient's systolic vital sign field is active in Fig. 3c, as evidenced by the inverted rolling keys 56. Proximate the rolling keys 56 is a scroll bar 58 for illustrating in a bar format the current value of the active vital sign field (i.e., systolic) which is inverted to the current level (130).

On Page 15, please amend the second full paragraph as follows:

Fig. 3(d) illustrates a patient inquiry screen 74 selected from the main menu 38 (also referred to as the data set inquiry screen). The patient inquiry screen 74 displays a scrolling text window 78 which exhibits the currently selected patient's name (data set header) and those names alphabetically proximate thereto. Along one side of the scroll text window 78 is positioned a scrolling bar 80 which includes an identifier 82 designating the position of the currently selected patient's name within a master alphabetical list. The user may scroll through the patient list by contacting the scrolling bar 80 at a desired location therealong. The top and bottom of the scrolling bar 80 corresponds to the beginning and end, respectively, of the list of patient names stored within the handheld interface 8.

On Page 17, please amend the third paragraph as follows:

Fig. 15 illustrates the RAM section 35 of the memory within the handheld interface 8. The RAM memory 35 includes a patient information section 200 for storing a list of patient names (data set headers) and patient identifiers (data set identifiers) associated with the present user of the handheld interface 8. The RAM memory 35 also includes a working space 202 for storing all other information entered by the user and transmitted between the handheld interface 8 and the communications server 12. Each time the CPU 32 transmits a request to the communications server 12 for patient information, the CPU 32 redefines the current data structure within the working space 102 to correspond to the expected format of the return data from the communications server 12.

On Page 18, please amend third and fourth paragraphs as follows:

Figs. 4-8 illustrate the processing sequence by which the CPU 32 controls the main menu, vital sign [patient] patient/data set input screen and the patient information screen (Figs. 3a-3d).

Generally, during operation the CPU 32 loops through one of two primary case/event handling loops (Fig. 4). The first case/event loop operates to draw the main menu (Fig. 3a) and to handle events chosen from the main menu. When an event occurs which corresponds to a defined key region within the main menu, a corresponding event identifier is returned as the calling identifier. A return code is set equal to this calling identifier and the return code is checked against a predefined value (0). If the return code is non-zero, processing flow enters the second primary loop to call the corresponding event handling function. The second primary

loop corresponds to processing in which a menu other than the main menu is displayed (Figs. 3b-3d). During this second loop, a code which identifies the next function to be performed is continuously checked. When the code equals zero, processing returns to the main loop. When the code is a nonzero value, the corresponding function is called. Once each function is completed it returns a code identifying the next function to be performed by the user. This configuration reduces the memory requirements by reducing the levels of functions to be called, thereby reducing the necessary stack space.

On Page 19, amend the first paragraph as follows:

Fig. 4 generally illustrates the processing undergone by the handheld interface 8 when the user initiates the first session (i.e., when the user first logs in). When a session is initiated the handheld unit prompts the user for the user's ID and password (step 1602). Next, a packet is constructed in the temporary buffer 204 which contains a long header 206 structure and [having] has a data section including the user ID and password (step 1604). This packet is transmitted (step 1606) and a validation code therefor is waited upon. If the validation code is not received (step 1608) processing returns to step 1602 in which the user is reprompted for the ID and password. If a validation code is received, processing continues to step 1610 in which a return code is set to indicate that processing should move to the patient inquiry module. Thereafter, the return code is tested to determine whether it equals zero (step 1623). If the return code equals zero, processing returns to step 1614 in which the main menu is redrawn.

On Page 21, amend the first paragraph as follows:

Figs. 5 and 6 illustrate the process undergone when the patient information screen 74 is selected to be displayed (such as during the wake up function or when called by the user). First, an initialization routine is called in step 1700 (steps 1710-1720 in Fig. 6). This initializing process begins by determining whether the handheld interface 8 includes a list of patient names (data set headers) and IDs (step 1710). If this list does not exist (such as when the handheld interface has initially been woken up), the unit constructs and transmits a packet to the communications server 12 requesting the patient list associated with the currently signed-in user (step 1712). The communications server 12 receives this packet, identifies the user ID, and requests the appropriate information from the command server 14. Thereafter, the appropriate database 16 is read and the patient list for the user is transmitted back to the handheld interface 8. After the interface 8 sets up the data structure in the patient memory 200 for the patient list (step 1714), it waits for the returned patient list (step 1716). Once the patient list (name and ID) is received, it is stored in the patient memory 200.

On Page 22, please amend the first full paragraph as follows:

The event identifier is passed to a case/event statement which includes every possible valid value for the event identifier. By way of illustration, processing will continue along one of six possible processing paths to block 1724, 1726, 1728, 1730, 1750 or [1750] 1756, depending upon which event occurred. When event 1724 occurs (i.e., the user presses the escape icon), the system returns a zero calling identifier to the main loop (thereby indicating that no new event handling function has been selected). If the event indicates that the user has touched the scrolling bar 80, processing flows to box 1726 in which the system determines the exact location of the event (step 1732). This location is correlated with a pointer that is used as an index into the patient list to identify the new patient to be displayed in the scrolling text window 78. Once the event location is determined, the pointer into the patient list is updated and the scrolling text window 78 is redrawn to display the name of the selected patient and a limited number of patient names surrounding the selected name (step 1734).

On Page 23, amend the second paragraph has follows:

If the keypad 76 is touched, processing flows to block 1730, at which the letter is identified which corresponds to the region touched (step 1742). This selected letter is added to a temporary patient searching string within the work space memory 102 (step 1744). This letter is added to a search string (which is empty until the first letter is selected). Thereafter, the processor conducts a search based upon the search string into the patient list to identify the name most closely corresponding to the letter(s) selected from the virtual keypad 76 (step 1746). If a search string exists (i.e., the user has already entered some letters) the newly selected letter is

concatenated onto the search string and a new search is conducted upon the text strings within the patient list to find the first text string which is greater in alphabetic value than (i.e., closest to) the search string. The pointer is set to the closest patient name and the scrolling text window 78 and the scrolling bar 80 are updated (steps 1746 and 1748). If the user wishes to delete a letter from the search string, he/she simply [pressing] presses the delete region key.

On Page 25, amend the first paragraph as follows:

Figs. 7a, 7b and 8 illustrate the processing sequence undergone by the handheld interface 8 when the [vitals] vitals/data [input] input/output handling function is chosen (i.e., when a user desires to enter patient vitals). First, the display screen is cleared (step 1800) and the vitals (data I/O) format is drawn upon the screen (step 1802). Similarly, the key regions are defined which correspond to each even identifier. Next, it is determined whether the workspace memory contains vitals for a selected patient (step 1804). If not, the processor constructs and transmits a packet (step 1806) requesting patient vitals. Thereafter, the workspace memory 202 is set up in the data structure corresponding to patient vitals (step 1808). Thereafter, the handheld interface waits for the returned patient vitals, receives these vitals in one or more packets and disassembles the packets and stores the patient vitals in the workspace (step 1810). Next, the processor draws the patient vitals onto the screen (step 1812) and sets the default mode to a predetermined field (e.g., systolic field). The vitals for the default field are drawn into the scroll bar (step 1814) and the default rolling keys 56 corresponding to the default field are inverted. Thereafter, the handheld interface waits for an event to occur (step 1816 in Fig. 7a) and when it occurs, identifies the event number corresponding thereto.

On Page 30, amend the first and third paragraphs as follows:

Once the user enters the desired data and wishes to send this data to the communications server 12, the user presses the transmit function button 70. Once the CPU 32 identifies that an event has occurred which corresponds to the transmit function button 70, the main loop (Fig. 4) calls the transmit handling function. Fig. 4 illustrates the process by which the CPU 32 transmits data to the communications server 12.

When transmitting a message the first packet/first frame thereof is constructed with a header section 304 formed in a long header structure followed by a data field 306. The long header 304 includes a 4 byte command field 308, a 6 byte user identifying field 310, and a 4 byte message total field 312. The command field 308 identifies the process to be [formed] performed upon the subsequent data, the user ID identifies the user signed into the handheld interface 8 transmitting or receiving the packet 308 and the message total 312 identifies the total length of the message which will follow. This total length includes all bytes within subsequent packets 300 corresponding to this specific message 302. Thereafter, a data field 314 (having 114 bytes in a packet with a 128 byte structure) follows.

On Page 31, amend the first paragraph as follows:

If the message includes more data than will fit in a single packet, subsequent packets/frames are transmitted. These subsequent packets/frames are constructed with a short header structure 316 preceding the data segment 318. The short header 316 includes a 4 byte command 320 and a 4 byte positioning packet 322 identifying number to enable the receiving device to determine the position of the packet within the overall message. Packets containing a short header 316 includes a 120 byte data field for a packet formed with 128 bytes. During transmission, the first packet of each message includes a long header 304 structure followed by a data field, with each subsequent packet within the message including a short header 316 [instructed] structure followed by a data field. In this manner, the device is able to increase the amount of data transmitted within each packet for long messages. The transmitting device need not send the user ID and the message total more than once for a given message since the receiving device is able to associate corresponding packets with a single message 302 based on the packet number 322 and communication channel.

On Page 32, please amend the first full paragraph as follows:

To transmit a message (Fig. 10), the CPU 32 reads the current data from the workspace memory 200 in the RAM 35 (step 400). Next, the CPU 32 determines the length of the message and the ID of the user signed in to that handheld interface 8 (step 402). The CPU 34 constructs a packet header formed with the long header structure (step 404). The CPU 34 clears the packet number (step 406) and writes the data to the transmit buffer (step 408). If the packet is full (step 410), the CPU 32 transmits the packet and clears the buffer (step 412). If the packet is not full, it determines if more data exists to write to the buffer (step 414). If no more data exists, it transmits the packet. If more data exists, it [against] again writes to the packet. In step 416, it is determined if more data exists, and if not it [exists] exits. If so, the packet number is incremented (step 418). Next, the CPU 34 constructs a partial short header for the second frame to be transmitted in this message (step 420). The partial header includes the code for the corresponding command and the current packet number. Thereafter, the next segment of data is written to the packet (step 408) and steps 410 through 420 are repeated. Once the last packet is transmitted (step 414) and it is determined that no more data remains to be written to the packet (step 416), the system exits the transmit routine (step 422).

On Page 33, before the first full paragraph, please insert the title "Communication Server" and amend the first and second paragraphs as follows:

Fig. 16 illustrates a block diagram of the communication server 12. The communication server 12 includes an IR receiver/transmitter 100 which receives and transmits IR communication packets from and to the handheld interface 8. The IR receiver transmitter 100 operates in conjunction with an I/O card 102 to store a packet of information from each communication channel in the corresponding address [with] within a temporary buffer 104. Each communication channel corresponds to a unique handheld interface 8. For instance, the communication server 12 may provide for 128 IR channels and thus, the temporary buffer 104 will include 128 buffer locations, with each buffer location having sufficient memory to store a complete IR packet 300 (Fig. 9). The number of channels is locally definable and may be any desired number. The array locations within the temporary buffer 104 store the IR packets in the format transmitted from the handheld interface 8 as described above.

The [temporary] communications server 12 further includes an input buffer 108 which represents a storage space used by the CPU 106 when converting packets from the format stored in the temporary buffer 104 to message lists with a different format to be transmitted to the command servers 14. The input buffer 108 represents an array, each element of which includes the COM_INFO structure (explained below).

On Page 36, please amend the first and second paragraphs as follows:

The command (CMD) represents the command to be processed in accordance with the corresponding message being transmitted to or from the handheld interface 8. The 4 bytes within the command are separated such that the first two bytes identify the database to be processed when performing the command, the third byte stores a number corresponding to the specific command to be performed, and the fourth byte is reserved. By way of example, with respect to the third byte, numerals 1-127 represent commands to be processed by the command server, while numerals 128-255 represent [to] a command from the handheld interface 8. As noted above, the use of shorthand code numbers for specific commands reduces the amount of data to be transmitted to and from the handheld interface 8.

The Packet Number is a long integer which is incremented each time a packet is appended to a message on the input buffer 108. The User_ ID_To is a 6 byte value added by the handheld interface 8 to identify a destination user. If set to zero, the destination is the communications server. The communications and command servers determine which server to send the command to. This value if non-zero will identify the user of a handheld interface 8 [desires] desiring to communicate therewith. The User_ID_From is a 6 byte value added once at login time. This user ID is assigned to the channel to identify who is logged in at that channel. This value is maintained until the person signs out. Thus, the user ID will only be transmitted once during a login session. The communications server 12 keeps track of each User ID signed onto handheld interfaces served by that communications server 12. The message total length (MSG_ Total) is a value assigned by the handheld interface 8 or by the command server 14 when a message is transmitted. The message length (MSG_LEN) is a value updated by the short and long header parsing functions to keep track of the length of a message in the message buffer 110

as the packets for the message [is] are added thereto. The message length field is initially tested by the communications server 12 when processing each packet in the temporary buffer 104 to determine if the packet is in a long or short header form. The message pointer field (MSG_ From_ HH) is a pointer into the message buffer to the location of the actual data message. The MSG_ From_ HH pointer is updated each time a new packet is appended to a message.

On Page 38, amend the first paragraph as follows:

[Figure 11] Fig. 11 illustrates the processing sequence by which the communication server 12 receives messages from the handheld interface 8, processes these messages and transmits these messages to the command server 14. First, the I/O card 102 is initialized, along with the communication channels, buffers and queues (step 700). Next, the data structures for the message list, COM_INFO, CMD, and short and long headers are set up (step 702). A current channel to be read by the I/O card 102 is initialized to the first channel (step 704). Next, the current channel is tested to determine whether data is being transmitted thereon from the corresponding handheld interface 8 (step 706). If data is present, the packet of transmitted data is read and stored in the temporary buffer 104 (step 708). The packet of data is stored within the temporary buffer 104 at the array address corresponding to the current channel. A channel identifying number corresponding to the transmitting handheld interface 8 is also assigned thereto by the I/O card 102.

On Page 39, amend the second paragraph as follows:

Fig. 12 and Fig. 13 illustrate the parse long and short header functions. Within the parse long header function, the first four bytes of the data packet within the temporary queue is read as the command (step 722). This command is compared with a list of valid commands and if the command is invalid the processor takes the necessary corrective action (step 724). If the command is valid, the command is written to the command field within the element of the input buffer 108 indexed by the corresponding channel number (step 726). Next, the subsequent six bytes of the packet within the temp buffer is read as the [user] User ID, if present, of the destination user for the attached message/data. This six byte ID is stored in the [user] User ID_To field of the element within the input buffer 108 indexed by the current channel number (step 728). The User ID, corresponding to the signed on user, is added to the UserID_From field of the indexed element.

On Page 42, please amend the third full paragraph as follows:

Next, processing moves to the processing queue sequence (Fig. 14) to move completed messages 302 to the processing queue 112. First, the current element pointer into the input buffer 108 is set to the first input buffer element (step 800). Then, it compares the message total length (MSG_Total) with the message length (MSG_LEN) to [determined] determine whether a complete message corresponding to the current element within the input buffer 108 has been received and stored in the message buffer (step 802). If not, the incomplete message remains on the input buffer 108 and the current element pointer is incremented (step 804) and the process returns to step 802.

On Page 43, please amend the first full paragraph as follows:

If the complete message for the current element has been received, the message is placed on the processing queue (step 806) by creating a message list therefor. To do so, the command information (COM_ INFO) of the current element of the input buffer 108 is written to the processing queue 112 and stored in the command information (COM_ INFO) section therefor. As the corresponding message represents a transmission from the handheld interface 8 to the command server 14, the next four fields of the indexed message list element in the processing queue remain undefined. This message list is "pushed onto the back" of the processing queue by storing, in the preceding message list in the processing queue, a pointer to the current message list. This pointer is [stored] stored in the final field (Previous_ MSG_ LST) within the previous, most recent message list added to the processing queue (step 806).

On Page 47, please amend the first full paragraph as follows:

Fig. 19 illustrates the process by which the command server 14 invokes the database management functions. First, the command server 14 [receives the] receives a message (step

900), including the complete message list structure followed by the data base file field (step 902) and determines whether that data base code is valid for the present command server 14 (step 904). If not, it returns an error message (step 906). If so, control is transferred to the database routine corresponding to the command code (CMD) in the Command field of the message list (step 908). Next, it is determined whether the command code is correct for the addressed database (step 910). If not, it returns an error code (step 906). If so, it accesses the message data within the message list and performs the appropriate operation upon the data base (step 912). Thereafter, it constructs a return message using the data returned from the database into the structure of the message list (step 914) and quits (step 916).

On Page 49, please amend the definition of "vital_input" as follows:

vital_input - contains all the functions used to perform the vital input
note - the Mode value indicates what is being input at the moment, if the user has [chose] chosen systolic, then the systolic rolling keys are inverted and activated, and the scroll bar will have the systolic data in it.

On Page 51, amend the definition of "place_message" as follows:
place_message - this routine will determine if the message being read [in] is the first or subsequent message. If the message is the first packet, then the routine parse_long_hdr is called. If the packet is the second or subsequent message, the routine parse_short_hdr is called.

On Page 52, please amend the definition of "process_in_buffer" as follows:
process_in_buffer - this routine is used to search the in-buffer for completed messages to [e] be placed on the process queue. This routine determines if a [messages] message is finished if msg_total =-msg_len and msg_total > 0. This routine calls ENQUEUE to physically put the message on the process queue.

please amend the definition of "packet_msg" as follows:
packet_msg - this routine is used to physically send the message from the communication server to the handheld computer via the AndroDat Card. It is responsible for all [packet] packets of the data and verify of successful transmission.

please amend the definition of "dequeue" as follows:
dequeue - this routine will take messages off the queue and place them in a temporary buffer to be processed by the communication server. A parameter is passed to which queue to process along with the temporary space to put message.

please amend the definition of "in_buffer" as follows:
in_buffer - this is an array of [teh] the structure com_info. This structure is used to hold the individual messages being received from the handheld computers.